Постановка задачи:

1) Реализовать функцию *nash_equilibrium(a)*, принимающую матрицу выигрышей и возвращающую оптимальные стратегии обоих игроков и цену игры.

2) Визуализировать спектр оптимальных стратегий с помощью *Jupiter Notebook* для игр с:

а) полным спектром оптимальных стратегий

б) неполным спектром оптимальных стратегий

в) спектром, состоящим из одной точки

Инструкция по запуску:

1) Установить библиотеки *Numpy, Scipy, Matplotlib* а так же *Jupyter Notebook*

2) Открыть файл *task2.ipynb* в *Jupyter Notebook*

Подход к решению задачи и что было использовано:

1) Для реализации *nash_equilibrium(a)* была использована функция *x = linprog (f, A, b)* из библиотеки *SciPy,* которая в зависимости от параметров решает разные задачи линейного программирования. В нашем случае она находит такой x, что $f^T * x \rightarrow min$, при условии, что $A * x \leq b$, где A - матрица выигрыша, x – возвращаемый функцией вектор. Наша задача сводится к преобразованию исходных данных, которые принимает функция *linprog,* таким образом, чтобы уместно было её использовать в рамках алгоритма поиска оптимальных стратегий. Переменная *mina* создана для того, чтобы исходную матрицу привести к другой матрице такой, что ее значение игры всегда будет положительным. Это делается для предотвращения ошибок при использовании функции *linprog.* Затем, обратными преобразованиями мы выводим результат для исходной матрицы. Этот подход справедлив в силу теоремы об аффинных преобразованиях.

2) Для визуализации спектров оптимальных стратегий была реализована функция *visualization(p)*, где p - вектор оптимальной стратегии. В ней используются функции *pyplot.axis, pyplot.scatter, pyplot.axvline* и *pyplot.show* из библиотеки *Matplotlib* (создание осей, установка точек, проведение вертикальных линий от этих точек до оси и вывод самого графика соответственно). А также был введён массив цветов *colors,* и с помощью функции *random.shuffle(colors)*, которая случайно перемешивает этот массив при каждой новой генерации, отображаются точки случайных цветов для более красивой картинки.

3) Для удобства тестирования была создана функция *run(a)*, принимающая матрицу a. В *run(a)* вызываются функции *nash_equilibrium(a), visualization(p)* и выводятся необходимые данные.